Available online at www.sciencedirect.com
ScienceDirect
International Journal of Approximate Reasoning
47 (2008) 109–124
INTERNATIONAL JOURNAL OF APPROXIMATE REASONING
www.elsevier.com/locate/ijar

Distance measures for biological sequences: Some recent approaches

Sabrina Mantaci, Antonio Restivo *, Marinella Sciortino

University of Palermo, Dipartimento di Matematica ed Applicazioni, Via Archirafi 34, 90123 Palermo, Italy

Received 20 April 2006; received in revised form 5 September 2006; accepted 15 March 2007
Available online 11 April 2007

Abstract

Sequence comparison has become a very essential tool in modern molecular biology. In fact, in biomolecular sequences high similarity usually implies significant functional or structural similarity. Traditional approaches use techniques that are based on sequence alignment able to measure character level differences. However, the recent developments of whole genome sequencing technology give rise to need of similarity measures able to capture the rearrangements involving large segments contained in the sequences. This paper is devoted to illustrate different methods recently introduced for the alignment-free comparison of biological sequences. Goal of the paper is both to highlight the peculiarities of each of such approaches by focusing on its advantages and disadvantages and to find the common features of all these different methods.

1. Introduction

The problem of comparing two sequences has recently assumed a fundamental importance in computer science, mainly motivated by problems regarding the structure of biological sequences such as DNA and proteins. Both of these biological structures can in fact be represented by sequences over a given alphabet: the nucleotides alphabet $\{A, C, G, T\}$ for DNA and the amino-acid 20 letter alphabet for proteins. The reason of it is quite simple. It is believed and proved that structural similarities between genomic sequences correspond to similar features and functionality of the structures that they represent. The urgency of defining good similarity measures between strings has grown since the possibility of sequencing the whole genome has raised the question of discovering common features between biological sequences of different species, reflecting on common evolutionary and functional mechanisms. This reason has led researchers to look for a definition of a distance measure on sequences able to capture these common mechanisms. Most of the traditional

* Corresponding author.
E-mail addresses: sabrina@math.unipa.it (S. Mantaci), restivo@math.unipa.it (A. Restivo), mari@math.unipa.it (M. Sciortino).

0888-613X/$ - see front matter © 2007 Elsevier Inc. All rights reserved.
doi:10.1016/j.ijar.2007.03.011

methods for comparing biological sequences were based on the technique of sequence alignment (cf. [17]). Nevertheless, sequence alignment considers only local mutations of the genome, therefore it is not suitable to measure events and mutations that involve longer segments of genomic sequences. For this reason many alignment-free distance measures have been recently introduced (see [39] for a survey).

In this paper we survey some recent approaches used to define alignment-free distance measures. We also would like to highlight advantages and disadvantages of these approaches and to find the common features of all these different methods for comparing sequences.

Some of them are based on counting the factors frequencies, and are recalled in Section 2. The basic idea underlying such a definition is that the more similar two sequences are, the greater it is the number of the factors shared by the two sequences.

An other group of distance measures, recalled in Section 3, are based on data compression (cf. [23,10,2,3,22]). The intuitive idea is that the more similar two sequences are, the more effective their joint compression is than their independent compression.

A third intuition, presented in Section 4, generalizes somehow the definition of sequences alignment, where the basic edit operation used in sequence alignment are integrated with a new set of operations. Then in this new approach, the allowed operations are: *character edits*: character insertions, deletions and substitutions, and *block edits*: block copying, deletion and relocation. The similarity or dissimilarity of two sequences is deduced by computing the minimal number of edit/block-edit operations needed in order to transform one sequence into another. Although block-edit distance seems to be a good distance definition on biological sequences, unfortunately it is proved that computing the block-edit distance of two words is NP hard.

Therefore a new framework has been developed (cf. [15]) in order to approximate such a distance. Then in Section 5 we report a set of distance measures between sequences that are very efficient to compute and that are based on the well known Lempel–Ziv parsing algorithm. Notice that, in spite such distance measures do not explicitly apply compression to the input sequences, they are closely related to the compression based distances of Section 3, since the parsing technique involved in their definition is a preprocessing to the LZ77 data compression method. Moreover we show that such Lempel–Ziv based distances provide the best known approximation of the block edit distance. This stresses the relevance of such distance measures since, beyond the efficiency of their computation, they state a formal bridge between the compression based distances and the block edit distance.

In the last section of this paper we introduce a different approach to sequences comparison. This method is combinatorial by nature and does not use any compressor, but, like the distance measures defined in the previous section, it makes use of a preprocessing of another well known compression algorithm (cf. [7]). Somehow, the needed information for comparison is already contained in the output of the preprocessing phase. Such a method is based on an extension of the Burrows–Wheeler Transform (`bwt`), a transformation widely used in the context of Data Compression. The new transformation introduced in [27] and denoted by `ebwt`, works analogously to `bwt`, but takes as input a multiset S of sequences. Actually we define a family of distances associated to the string produced by the `ebwt`, since we can give a different distance measure in correspondence of each possible parsing of such a string. As for the k-tuple count Euclidean distance, the `ebwt`-based distance measures take into account that the more similar two sequences are, the more factors they share. This information can be detected by looking at the output of the transformation. The connection with the k-tuple count Euclidean distance is even more effective. In fact we can always find an `ebwt`-based distance measure that approximates the k-tuple count Euclidean distance.

The goal of this paper is to illustrate different methods recently introduced for the comparison of biological sequences. The formal proofs of the theorems here reported and the experimental validation of the different methods can be found in the references.

2. Distances based on word frequencies

One of the oldest approaches for sequences comparison is based on the frequencies statistics of the length k factors (cf. [5]). These methods transform a sequence into an object on which the tools commonly used in Linear Algebra and Statistical Theory can be applied. The main idea of such an approach is based on the

remark that similar sequences share common factors; therefore these methods are based on a "count" of the number of the factors of a given length shared by the compared sequences. Even if such a method is alignment-free, it is still length dependent in the sense that the comparisons are made for a fixed factor-length. Actually, such an approach can be thought as a weak starting point for an alignment because, intuitively, it is equivalent to recognize local alignments between identical segments of sequences. In fact in [6] it is proved that the dissimilarity values observed by using distance measures based on word frequencies are directly related to the ones requiring sequence alignment. Moreover in [35] it is remarked that the distances based on word frequencies as a filtration method for sequence alignment algorithms are good tool for increasing the efficiency of such algorithms. This is because the elimination of the sequences with a low similarity level allows to reduce the input to the dynamic programming algorithms for sequence alignment, having quadratic complexity. In fact, many heuristics for fast database similarity search in molecular biology use such a filtration approach. Both FASTA [[34]] and BLAST ([1]), for instance, preselect candidate sequences for alignments by starting from short identical or highly similar fragments.

Nevertheless in [21] it is shown that alignment-free distances based on the count of k-tuples are not interesting in order to construct a phylogeny. In fact experiments for constructing phylogenetic trees directly from such a kind of distance lead to an odd classification.

In general, since the comparison methods based on the frequencies of the factors transform a sequence into an integer vector, any of the distance defined on the vector space $\mathbb{N}^m$ (for an appropriate m) can be applied in order to define a distance measure between sequences. Here we refer to the most classical of the distances over a vector space, that is the Euclidean distance of order p. This gives rise to the definition of the k-tuple count Euclidean distance on the set of sequences over A^*.

Given a sequence u over the alphabet A, we define r_u^k as a vector of dimension $|A|^k$ over $\mathbb{N}$, such that for each index $i = 1, \ldots, |A|^k$, $r_u^k[i]$ counts the number of the occurrences in u of the ith word of A^k, taken in lexicographic order. In this way a vector $r_u \in \mathbb{N}^{|A|^k}$ is associated to each sequence u. The basic idea for comparing two sequences u and v is that we can associate to them the corresponding integer vector in $|A^k|$ components, and apply any of the distance measures defined over $\mathbb{N}^{|A|^k}$. In particular, for any integer p, we can apply the Euclidean distance of order p. More formally, let u and v be two sequences over A^*. Then the *k-tuple count Euclidean distance of order p* is defined as:

$$D_k^p(u,v) = \sum_{i=1}^{|A|^k} |r_u^k[i]^p - r_v^k[i]^p|^{1/p}.$$

In particular for $p = 1$ we have:

$$D_k(u,v) = \sum_{i=1}^{|A|^k} \left| r_u^k[i] - r_v^k[i] \right|.$$

Usually the integer k is called *resolution*.

The following example show how this distance is computed on two sequences.

Example 1. Consider the sequences $u = abaababb$ and $v = baabbaba$ and let $k = 3$. The lengths three words over the alphabet $\{a, b\}$, listed in the lexicographic order are: $\{aaa, aab, aba, abb, baa, bab, bba, bbb\}$. Then the vectors on $\mathbb{N}^8$ corresponding to u and v are respectively: $r_u^3 = (0,1,2,1,1,1,0,0)$ and $r_v^3 = (0,1,1,1,1,1,1,0)$. Then $D_3(u,v) = 2$. If $k = 4$ then we have to consider a vector with 16 components. It is easy to verify that $D_4(u,v) = 8$.

We can note that the vectors obtained for the computation of the distance represent the original sequences with a fixed resolution k that is the considered factor-length. The effectiveness of such an approach is closely related with the identification of optimum values for the factor-length k (optimal resolution). Such an optimality depends both on the features of the sequences to be compared and on the algorithm that is eventually based on this distance. A related approach consists of combining results obtained with arbitrary factor-length intervals. In following sections we describe some methods that are resolution-free, i.e. that do not need to fix "a priori" a resolution for comparing sequences.

3. Distances based on compression

This section is devoted to describe a class of methods that make use of some text compression techniques for comparing sequences. Such methods are based on the general idea that the more two sequences are similar, the more succinctly we can describe one given the other. In terms of compression, this means that two sequences are considered close if one sequence is significantly compressible given the information contained in the other. In particular, the distance here described is computed from the lengths of compressed sequences singly and in pairwise concatenation. A pioneer work in that direction is [2], in which a very general method for extracting information from a generic string of characters is given. In this paper the distance between two texts is defined as the relative entropy between the two sources from which the texts are produced. In order to measure the relative entropy the LZ77 compression algorithm has been used (cf. [19]). The method has been applied to a linguistic motivated problems, such as language recognition, authorship attribution and language classification.

In [10] a first theory of real-world compressor-based normalized compression distance is proposed. A theoretical precursor of such a distance is a similarity metric introduced in [23] that is based on a notion from the Kolmogorov complexity theory. The Kolmogorov complexity of a string x given the string y is the length of the shortest binary program that on input y outputs x on an appropriate universal computer, such as a universal Turing machine. Such a complexity is denoted by $K(x|y)$. For more details, see [24]. The Kolmogorov complexity of a string x is defined as the length of the shortest binary program that outputs x without input, i.e. $K(x) = K(x|\epsilon)$. Intuitively, $K(x)$ is the length of the ultimate compressed version of the string x. In [3] was introduced the so-called *Information Distance* $E(x, y)$ defined as the length of the shortest binary program on a universal Turing machine that computes y with input x and computes x with input y. It was shown that, up to an additive logarithmic term,

$$E(x, y) = \max\{K(x|y), K(y|x)\}.$$

It was proved also that E is a metric, up to a negligible violations of the metric inequalities. In [23] a normalized version of E is defined as:

$$\mathrm{NID}(x, y) = \frac{\max\{K(x|y), K(y|x)\}}{\max\{K(x), K(y)\}}$$

and it is called Normalized Information Distance.

It was proved that NID is also a metric and is universal in the sense that it minorizes, up to a negligible additive error term, a class of so-called *normalized admissible distances* (cf. [23,10]). This means that if two sequences are similar according to a particular feature described by a particular admissible distance (not necessarily a metric), then they are similar according to the NID metric.

Since this metric is based on the Kolmogorov complexity, it is not computable. So, by using an additive property of Kolmogorov complexity, given a compressor C it was defined an approximation of the previous distance defines as:

$$\mathrm{NCD}(x, y) = \frac{\min\{C(xy), C(yx)\} - \min\{C(x), C(y)\}}{\max\{C(x), C(y)\}}$$

that is called *normalized compression distance*. One can notice that if the compressor C is block-coding based (cf. [7]), then it is symmetric, so the previous definition becomes:

$$\mathrm{NCD}(x, y) = \frac{C(xy) - \min\{C(x), C(y)\}}{\max\{C(x), C(y)\}}.$$

In practice, the NCD is a non-negative number r, such that $0 \leqslant r \leqslant 1 + \epsilon$ representing how different the two strings are. The ϵ in the upper bound is due to imperfections in the compressor, but for more standard compression algorithms one is unlikely to see an ϵ above 0.1.

Let us consider now a class of compressors that includes most real-world compressors and assure that the NCD is a metric.

A compressor C is called normal if it satisfies, up to an additive logarithmic term, the following:

(1) $C(xx) = C(x)$ and $C(\epsilon) = 0$;
(2) $C(xy) \geqslant C(x)$;
(3) $C(xy) = C(yx)$;
(4) $C(xy) + C(z) \leqslant C(xz) + C(yz)$.

It was proved in [10] the following result.

Theorem 2. *If the compressor C is normal then the NCD is a metric.*

Moreover NCD is called a *quasi-universal* distance because it minorizes every computable normalized admissible distance up to an error depending on the quality of the compressor's approximation of the Kolmogorov complexities of the sequences.

Earlier approaches can be found also in [22] in which a first completely automatic construction of the phylogeny tree based on whole mitochondrial genomes is shown, in [8] where a plagiarism in student programming assignment is automatically detected, in [4] in which a phylogeny of chain letters is given and in [11] where an automatic clustering of music files is obtained. The method described in [10] has been implemented and it is available as public software (cf. [9]). It has been verified that it also robust under choice of different compressors. In [10] successful application in genomics, virology, linguistic, literature, music, handwritten digits, astronomy and combinations of objects from completely different domains are reported. In such experiments very different type compressors has been used, as statistical (PPMZ), Lempel–Ziv based dictionary (gzip), block based (bzip2) and special purpose (Gencompress). Nevertheless the robustness of the method shows also some inadequacy of the method itself both to use the specific features of each compressor, and to disclose the particular type of similarity between the considered sequences.

4. Block edit distance

In this section we consider a notion of similarity between two words described in terms of a *transformation* of one word into the other through a set of specified operations. A classical, and simple, formalization of this notion is the so called *edit distance*. It is based on the transformation (or editing) of one word into the other by a series of edit operations on individual characters. The permitted edit operations are *insertion* of a character into a word, *deletion* of a character from a word, or *substitution* (or *replacement*) of a character in a word with a character in the other word.

Definition 3. The edit distance between two words is defined as the minimum number of edit operations – insertions, deletions, substitutions – needed to transform the first word into the second.

Edit distance between two words x and y is denoted by $D_{\mathrm{L}}(x, y)$ and it is sometimes referred to as *Levenshtein distance* in recognition of the paper [21] by Levenshtein where edit distance was firstly discussed (cf. also [32,40,47] for a survey).

The *edit distance problem* is to compute the edit distance between two given words. By using dynamic programming, one can prove the following theorem (cf. [17]).

Theorem 4. *The edit distance $D_{\mathrm{L}}(x, y)$ can be computed in time* $\mathrm{O}(|x||y|)$.

With the advance of genomic sequence technologies we now have access to complete genome sequences of humans and many other species. As a result, the notion of similarity between genome sequences has changed dramatically in the last years. In order to capture evolutionary mechanisms underlying genome evolution one must consider a richer set of sequence modifications than single character edits. This has lead to new sequence comparison methods that capture similarity based on not merely individual nucleotides (i.e. characters) but involving longer segments of the genome. So, together with the character edit operations, one introduces also *block edit operations*, which involve moving a block (any consecutive set of characters) from one location to another, or copying a block from one place to another within a sequence, or deleting a block.

In a more formal way, with reference to the model considered in [15], the edit operations of interest are:

- *character edits*: character insertions, deletions and substitutions,
- *block edits*: block copying, deletion and relocation.

By a *block copying* we mean that a block (or factor) z occurring in a certain place in a word x is duplicated in another place of x. For instance, the word $y = abacbbacc$ is obtained from the word $x = abacbbc$ by copying the block $z = ac$, occurring in the position 3 of x, in the position 7 of x (by position of a block z in x we mean the index, or position, of the first character of z in x). By a *block deletion* we simply mean the cancellation of a given block z occurring in a word x. For instance, the word $y = ababc$ is obtained from the word $x = abacbbc$ by deleting the block cb. By a *relocation* we mean that a certain block z occurring in a word is moved from its place to another place in x. For instance, the word $y = abbbacc$ is obtained from the word $x = abacbbc$ by moving the block ac from its original place to the position before the last c.

Definition 5. The block edit distance from the word x to the word y, denoted by $D_B(x \to y)$, is defined as the minimum number of block edit operations needed to transform x to y.

The block edit distance is not necessarily symmetric, i.e. there may be words x and y for which $D_B(x \to y) \neq D_B(y \to x)$. The symmetric version of the block edit distance can be defined as $D_B(x,y) = \frac{1}{2}[D_B(x \to y) + D_B(y \to x)] = D_B(y,x)$.

The block edit distance considered in this section is close to other similarity measures, based on edit operations, proposed in literature. The key difference is in the set of edit operations that are allowed. In some cases block deletes are either not allowed at all (cf. [12]), or are allowed only in a restricted way. In particular, in [13,30] any deletion can be performed only in a block that has another copy elsewhere in the remainder of the word: in this way such a block delete operation is a sort of inverse of a block copy operation. In other approaches (cf. [31]) block copies are not allowed. In some other papers, in the set of allowed operations is included also the *block inversion* operation (cf. [38]).

The *block edit distance problem* is to compute the block edit distance between two words.

Theorem 6 [25]. *The block edit distance problem is NP-complete.*

In the following section we will consider some efficient approximations of the solutions of the problem.

5. Distances based on LZ parsing

In this section we consider some similarities measures between sequences that are very efficient to compute and that are based on the well known Lempel–Ziv parsing algorithm. Even if we do not make explicit reference to any compressor, such similarity measures are closely related to the compression based distances, since the parsing techniques involved in their definition play a central role in some data compression methods. On the other hand, in this section we show, following [15], that such LZ based distances provide the best known approximation of the block edit distance. This stresses the relevance of the similarity measures here introduced, since, beyond the efficiency of their computation, they state a formal bridge between the compression based distances and the block edit distance.

In [18] Abraham Lempel and Jacob Ziv introduced the following parsing algorithm, which is the base of their data compression methods (cf. [19,20]). We use the variant proposed in [17]. Given a word x, the Lempel–Ziv (LZ) parsing of x is

$$\mathrm{LZ}(x) = (z_1, z_2, \ldots, z_m, z_{m+1}, \ldots, z_n),$$

where $x = z_1 z_2 \cdots z_n$, the block z_m is the longest prefix of $z_m z_{m+1} \cdots z_n$ which occurs as factor in $z_1 z_2 \cdots z_{m-1}$, if such a prefix is different from the empty word, and z_m is the first character of $z_m z_{m+1} \cdots z_n$, otherwise. The integer n is called the *complexity* of the word x and it is denoted by $c(x)$.

Consider, for instance, the word $x = abaababaabaababaabab$. The LZ parsing of x is:

$$\mathrm{LZ}(x) = (a, b, aa, aba, baaba, ababaaba, b),$$

and the complexity of x is $c(x) = 7$.

In order to introduce a measure of relative entropy between individual sequences in [41] Ziv and Merhav describe a variant of the LZ parsing algorithm, which takes into account two word. Given two words x and y, the ZM parsing of y with respect to x is

$$\mathrm{ZM}(y|x) = (z_1, z_2, \ldots, z_m, z_{m+1}, \ldots, z_n),$$

where $y = z_1 z_2 \cdots z_n$, the block z_m is the longest prefix of $z_m z_{m+1} \cdots z_n$ which occurs as factor in x, if such a prefix is different from the empty word, and z_m is the first character of $z_m z_{m+1} \cdots z_n$, otherwise. The integer n is called the *complexity of y relative to x* and it is denoted by $c(y|x)$.

Consider, for instance, the words $x = abbababbaab$ and $y = abbbababbba$. The ZM parsing of y with respect to x is:

$$\mathrm{ZM}(y|x) = (abb, bababb, ba),$$

and the complexity of y relative to x is $c(y|x) = 3$.

It is important to note that, given two words x and y, the parsing $\mathrm{ZM}(y|x)$, and then also $c(y|x)$, is computable in $O(|x||y|)$ time via the use of (on-line) suffix tree construction algorithms (cf. [36]).

The notion of ZM parsing of one word with respect to another is closely related to the following factorization problem. Given a finite set F of words, and a word y, a factorization of y in elements of F is a sequence of words $z_1, z_2, \ldots, z_n \in F$ such that $y = z_1 z_2 \cdots z_n$. The *optimal factorization problem* is to find a factorization such that n is minimal. The simplest strategy to factorize y is to use a greedy algorithm. The factorization is computed sequentially. Therefore the first factor z_1 is naturally chosen as the longest prefix of y that belongs to F, and the decomposition of the remainder of the text is done recursively in the same way. Remark however that this greedy algorithm does not produce, in general, the optimal factorization. Consider, for instance, the set $F = \{a, b, c, ab, ba, bb, bab, bba, babb, bbab\}$, and the word $y = aababbabbabbc$. The greedy algorithm produces the factorization:

$$y = a.ab.ab.babb.ab.b.c$$

that contains seven factors. This factorization is not optimal. The optimal factorization is

$$y = a.a.ba.bbab.babb.c$$

that contains six factors. However, one can easily prove (cf. [14]) the following lemma. Recall that a language L is *factorial* if it is closed by factor, i.e. if all of the factors of elements of L belong to L.

Lemma 7. *Il F is a factorial language and y is an arbitrary word, then the greedy algorithm produces the optimal factorization.*

Remark that the ZM parsing of a word y with respect to a word x corresponds to the greedy algorithm to factorize the word y in elements of $F(x) \cup A$, where $F(x)$ denotes the set of factors of the word x and A denotes the alphabet of y. As a consequence of previous lemma, we have the following proposition.

Proposition 8. ZM(y|x) *produces an optimal factorization of the word y in blocks of x.*

The notion of ZM parsing can be used to give a similarity measure between two words x and y. The idea is that, the greater is the number of factors shared by x and y, the smaller is the number of elements in $\mathrm{ZM}(y|x)$, i.e. the smaller is $c(y|x)$. In particular, if $x = y$, then $\mathrm{ZM}(y|x) = (x)$ and $c(y|x) = 1$. This suggest to define the *ZM distance* from a word x to a word y as follows:

$$D_{\mathrm{ZM}}(x \to y) = c(y|x) - 1.$$

For instance, if we consider the words $x = abbababbaab$ and $y = abbbababbba$, in a previous example, $D_{\mathrm{ZM}}(x \to y) = 2$.

Remark that this distance has the property that $D_{\mathrm{ZM}}(x \to y) = 0$ if and only if $x = y$. However this distance is not necessarily symmetric, i.e. there may be words x and y for which $D_{\mathrm{ZM}}(x \to y) \neq D_{\mathrm{ZM}}(y \to x)$. As in the case of block edit distance, the symmetric version of this distance can be defined as $D_{\mathrm{ZM}}(x, y) = \frac{1}{2}[D_{\mathrm{ZM}}(x \to y) + D_{\mathrm{ZM}}(y \to x)] = D_{\mathrm{ZM}}(y, x)$.

In [33] Otu and Sayood introduce another distance measure based on the Ziv–Lempel parsing. Given two words x and y, let $\mathrm{LZ}(y)$ be the LZ parsing of y and let $\mathrm{LZ}(x\#y)$ be the LZ parsing of the word $x\#y$, i.e. the concatenation of x and y separated by a delimiter # which is not a part of the alphabet of x or y. It is easy to see that the delimiter # corresponds to a block of this parsing, i.e. the parsing has the form

$$\mathrm{LZ}(x\#y) = (t_1, t_2, \ldots, t_r, \#, z_1, z_2, \ldots, z_k).$$

This means that $\mathrm{LZ}(x\#y)$ induces a parsing $(z_1, z_2, \ldots, z_k)$ of the suffix y. The number of blocks in such a parsing is $c(x\#y) - c(x) - 1$. Now, the idea is that, the closer are x and y, the smaller is number of blocks in such a induced parsing of y. In the extremal case of $x = y$, one has that $c(x\#y) - c(x) = 2$. Indeed, one has that $\mathrm{LZ}(x\#y) = (t_1, t_2, \ldots, t_r, \#, y)$.

This leads Otu and Sayood to define a distance, here called the *OS distance*, from a word x to a word y as follows:

$$D_{\mathrm{OS}}(x \to y) = c(x\#y) - c(x) - 2.$$

Consider, for instance, the words $x = abbababbaab$ and $y = abbbababbba$. One has:

$$\mathrm{LZ}(x\#y) = (a, b, b, ab, abba, ab, \#, abb, bababb, ba),$$

and then

$$D_{\mathrm{OS}}(x \to y) = c(x\#y) - c(x) - 2 = 10 - 6 - 2 = 2.$$

In this example $D_{\mathrm{OS}}(x \to y) = D_{\mathrm{ZM}}(x \to y)$. We have in general the following proposition.

Proposition 9. $D_{\mathrm{OS}}(x \to y) \leqslant D_{\mathrm{ZM}}(x \to y)$.

The proof of the previous proposition is based on the fact that the number of blocks in the parsing induced in y by $\mathrm{LZ}(x\#y)$ (i.e. $c(x\#y) - c(x) - 1$) is less than or equal to the number of blocks in $\mathrm{ZM}(y|x)$ (i.e. $c(y|x)$). Indeed, in the first case, a given block of y is determined by looking at the factors of x and also at the factors of the prefix of y already parsed. In the second case, the block is determined by looking only at the factors of x.

By definition, $D_{\mathrm{OS}}(x \to y) = 0$ if and only if $x = y$. However, as in the case of the distance $D_{\mathrm{ZM}}(x \to y)$, also this distance is not symmetric and a symmetric version can be defined. By using such a distance, Otu and Sayood have successfully constructed consistent phylogenies for real and simulated data sets (cf. [33]).

In the remainder of this section we show, following [15], that $D_{\mathrm{ZM}}(x \to y)$ and $D_{\mathrm{OS}}(x \to y)$ are approximations of the block edit distance $D_{\mathrm{B}}(x \to y)$.

The block edit distance between two words x and y is described in terms of a transformation from x to y through a set of edit operations. Any transformation that uses edit operations can be trivially emulated by another transformation which construct an initially empty word y' while using x only as source of blocks that can be copied into y'. It is then possible classify transformations from x to y and the distance between them according to the *source* and *destination* of edit operations permitted.

External transformations iteratively construct y from an initially empty word y' by performing edit operations whose source is x and whose destination is y'. Thus x does not change during the transformation. The corresponding distance is denoted by $D_{\mathrm{Be}}(x \to y)$ and is called the *external block edit distance*.

Internal transformations construct y by iteratively applying edit operations to word y' which is initially set to x. This is the general case and the corresponding distance is the block edit distance $D_{\mathrm{B}}(x \to y)$.

Remark that in internal transformations, the *source* (and the *destination*) is y', that changes at any step, whereas in external transformations the source is x and it does not change during the process. Remark also that$D_{\mathrm{B}}(x \to y) \leqslant D_{\mathrm{Be}}(x \to y)$.

Example 10. Consider the words $x = acbcabc$ and $y = abcbcbca$.

The destination word y can be constructed in 3 steps, starting from the empty word, by using *external* transformations: (1) $y' = ab$ (by the copy of ab from x); (2) $y' = abcbc$ (by the copy of cbc from x); (3) $y' = y = abcbcbca$ (by the copy of bca from x).

The destination word y can be constructed in two steps, starting from the word x, by using *internal* transformations: (1) $y' = acbcbca$ (by moving the right most a to the right end of $y' = x$); (2) $y' = y = abcbcbca$ (by copying b in second position of y').

In the sequel we show that external transformations are closely related to the Ziv–Merhav distance and the general (internal) transformations are closely related to the Otu–Sayood distance.

Let us first consider only *external* transformations. The set of allowed operations to build y from x includes three types of operations: (1) an operation which allows to create parts of y which are shared with x, called *copy* operations; (2) an operation which allows to construct part of y which are not shared with x, called *insertion* operations (in our case the only allowed insertion operation is the insertion of a single character); (3) an operation which allows to *delete* blocks in y. In the external transformations block relocations are not allowed.

Let us now consider two further restrictions on external transformations: (i) block delete operations are not allowed; (ii) one can use only *marginal* transformations, i.e. the block copy and character insertion are only allowed at the right or left end of the destination word.

By taking into account the above restrictions, the transformation builds the destination word y by adding blocks from the source x, one after the other, only on the left or on the right of the destination word. This means that y is built by *concatenating* blocks from x.

If $D'_{\mathrm{Be}}(x \to y)$ denotes the distance defined by external transformations with the above restrictions, then it is easy to see that $D'_{\mathrm{Be}}(x \to y)$ corresponds to the number of blocks in an optimal factorization of y in blocks from x. As a consequence of Proposition 8, one has that such a distance corresponds to the Ziv–Merhav distance.

Proposition 11. $D'_{\mathrm{Be}}(x \to y) = D_{\mathrm{ZM}}(x \to y) + 1$.

Let us now analyze the relationships between the *external block edit distance* $D_{\mathrm{Be}}(x \to y)$ and the distance $D'_{\mathrm{Be}}(x \to y)$ defined by introducing the above restrictions.

First remark that, if we delete a block B in y, such a block B must have been created in y through one or more block copy operations or character insertions. These operations, as well as the deletion of B can be emulated by only two block copy operations: one involving the part of y on the left boundary of B and the other involving the part on the right boundary of B. So the number of operations we need to construct y from x without using deletes is at most twice the number of operations we need using also block deletes.

Let us now suppose that we use only *marginal* transformations, i.e. the block copy and character insertions are only allowed at the right or left end of the destination word. The following example shows that such a restriction at most duplicates the number of operations needed to transform the source word x into the destination word y. Consider the source word $x = ab$ and the destination word $y = aaabbb$. y can be constructed via three block copies starting with the empty word and by copying in each step the block ab in the central position of the destination word. If the block copies are allowed only at the right end or left end of the destination word, then one needs six block copies, as the reader can easily verify. By the above arguments and the previous proposition we obtain the following approximation result.

Theorem 12. $D_{\mathrm{ZM}}(x \to y) + 1 \leqslant D_{\mathrm{Be}}(x \to y) \leqslant 4(D_{\mathrm{ZM}}(x \to y) + 1)$

The arguments to approximate the general (i.e. internal) case follow similar lines. Recall that $D_{\mathrm{B}}(x \to y)$ is the minimum number of steps to construct y by iteratively applying edit operations to word y' which is initially set to x. Consider also in this general case the above restrictive conditions: (i) block delete operations are not allowed; (ii) one can use only *marginal* transformations, i.e. the block copy and character insertion are only allowed at the right or left end of the destination word. Arguments very similar to those used for the external case show that the problem can be translated into the problem to determine the minimal parsing of the word y by using character insertions and copy of blocks coming both from x and from the part y' of y already parsed. This means that the block edit distance with the above restrictions, denote by $D'_{\mathrm{B}}(x \to y)$, is closely related to the distance define by Otu and Sayood. We can state indeed the following proposition.

Proposition 13. $D'_{\mathrm{B}}(x \to y) = D_{\mathrm{OS}}(x \to y) + 1$.

In order to state the relationships between the block edit distance $D_{\mathrm{B}}(x \to y)$ and its restrictive version $D'_{\mathrm{B}}(x \to y)$ one needs a more accurate and involved analysis than in the external case. This is done in [15], where the reader can find the detailed arguments for proving the following approximation result.

Theorem 14. $D_{\mathrm{OS}}(x \to y) + 1 \leqslant D_{\mathrm{B}}(x \to y) \leqslant 12(D_{\mathrm{OS}}(x \to y) + 1)$

In conclusion, the distance defined by Otu and Sayood in terms of the Lempel–Ziv parsing algorithm provides an approximation of the block edit distance. Moreover it is, to our knowledge, the best known approximation. On the other hand, since the Lempel–Ziv parsing is a fundamental step in some important data compression methods, the results of this section establish deep connections between compression based distance and the block edit distance.

6. Distances based on the BW Transform

In this section we deal with a new alignment-free method for comparing sequences, introduced in [27,28], which is combinatorial by nature and does not use any compressor nor any information-theoretic notion. Like the distance measures defined in the previous section, this method makes use of a preprocessing of a compression algorithm, but does not actually need to apply the compression. Experimental results show that there is a close relation between comparing sequences by this approach and by some compression based measures (cf. [7]). Somehow, the needed information for comparison seems to be already contained in what we get just after the preprocessing phase.

Such a method is based on an extension of the Burrows–Wheeler Transform (`bwt`), a transformation widely used in the context of Data Compression. Loosely speaking, `bwt` is a reversible transformation that produces a permutation $\mathtt{bwt}(w)$ of an input sequence w, and such that $\mathtt{bwt}(w)$ is much easier to compress than w (cf [7]). The new extended transformation, denoted by `ebwt`, works analogously to `bwt`, but takes as input a multiset S of sequences (for a systematical study of its properties see [29]).

In this section we are going to define a class of dissimilarity measures that, by using the `ebwt`, formalize the intuitive idea that the greater is the number of factors shared by two sequences u and v, the smaller is the "distance" between u and v. A fundamental step of the construction of the `ebwt` of a multiset S consists in sorting all the conjugates of the words in S. When applied to $S = \{u, v\}$, the conjugates of u and v are then mixed in such a sorted list. Notice that if the same factor s appears both in u and in v, then the conjugates of u and v starting by s are likely to be close in the sorted list. This implies that the greater it is the number of factors shared by u and v, the greater is the mixing of conjugates of u and v in the sorted list. The class of distance measure based on the `ebwt` will take into account such a mixing.

We here consider the transformation `ebwt` under the hypothesis that the words considered are primitive. This is not actually restrictive, since in practice almost all the processed texts are primitive (or become primitive by adding an end-of-string symbol). Moreover, for our application to sequences comparison, we just need to define the extended transformation to pairs of words (including the special case when the two words are equal), whereas in its general form, this extension is applied to any multiset of words. Therefore, in this paper the extended transformation will be directly defined on pairs of words. Moreover, we use here a special form of this extension, called the colored extended transform, that contains a supplementary output, essential in order to define a new notion of dissimilarity between words (cf. [26]).

In order to define the extended transformation we need to introduce an order relation between primitive words that differs from the usual lexicographical order when one word is a prefix of the other one. If u is a word in A^*, we denote by u^ω the infinite word obtained by infinitely iterating u, i.e. $u^\omega = uuuuu\cdots$. On infinite words it is naturally defined the lexicographic ordering, that is, taken two infinite words $x = x_1x_2\cdots$ and $y = y_1y_2\cdots$, with $x_i, y_i \in A$, we say that $x <_{\mathrm{lex}} y$ if there exists an index $j \in \mathbb{N}$ such that $x_i = y_i$ for $i = 1, 2, \ldots, j-1$ and $x_j < y_j$. Note that if $x = y$, the relation $<_{\mathrm{lex}}$ is not defined. Let u, v be two primitive words over a finite alphabet A. We say that:

$$u \preceq_\omega v \iff u^\omega <_{\mathrm{lex}} v^\omega$$

We remark that this order relation is different from the lexicographic one. In fact for instance $ab <_{\mathrm{lex}} aba$ but $aba \preceq_\omega ab$. Notice also that the $<_{\mathrm{lex}}$ and the $\preceq_\omega$ ordering coincide for those pairs of words such that none of them is a prefix of the other one. Moreover, although the $\preceq_\omega$ order of u and v is defined by using infinite words, the Fine and Wilf Theorem (cf. [16]) allows to show that it is possible to decide their mutual $\preceq_\omega$ ordering by extending them up to the length $|u| + |v| - \gcd(|u|, |v|)$. The Fine and Wilf theorem allows also to prove the tightness of this bound (see [29] for details).

Let (u, v) be a pair of (non necessarily distinct) words of A^*. Let us denote by R the red color and by B the blue color. The *coloring* of $\{u, v\}$ is the application γ defined $\gamma(u) = R$, $\gamma(v) = B$.

Recall that two words $x, y \in A^*$ are *conjugate*, if $x = uv$ and $y = vu$ for some $u, v \in A^*$. Let γ be the coloring of (u, v) and let $\mathscr{C}(u, v)$ denote the set of all the conjugates of u and v. The *extension of the coloring* γ *to* $\mathscr{C}(u, v)$ is defined as the application that for each $w \in \mathscr{C}(u, v)$ assigns to w the color R if w has been obtained as conjugate of u and the color B if w has been obtained as conjugate of v. Remark that even if u and v are equal or conjugates, they must have different colors. So, since u and v are primitive, if two equal words appear in $\mathscr{C}(u, v)$, they must be necessarily derived as conjugates of u and v, respectively, so they must have different colors. We use the same symbol γ for the extension of the coloring to $\mathscr{C}(u, v)$, where there is no danger of ambiguity.

We denote by $L(w)$ the last character of the word w.

Consider the set of triplets

$$\{(w, L(w), \gamma(w)) | w \in \mathscr{C}(u, v)\}$$

and sort it by taking as sorting key the first component, using the $\preceq_\omega$ order relation. This sorting is performed in a stable way with respect the third component, by considering $R < B$. We can arrange this sorted list in a table M with $|u| + |v|$ rows and 3 columns (see for instance Fig. 1). We denote by $M_{\mathscr{C}}$, M_{L} and M_γ the first, second and third columns, respectively, of the table M. Notice that $M_{\mathscr{C}}$ represents the sorted list of the words in $\mathscr{C}(u, v)$, according to the $\preceq_\omega$ order relation, M_{L} is the word obtained from the concatenation of the last characters of elements in $M_{\mathscr{C}}$ and M_γ is the sequence of the colors of the elements in $M_{\mathscr{C}}$. When it is useful to specify the pair (u, v), we will write $M_{\mathscr{C}}(u, v)$, $M_{\mathrm{L}}(u, v)$ and $M_\gamma(u, v)$ instead of $M_{\mathscr{C}}$, M_{L} and M_γ, respectively. Moreover we denote by $M_{\mathscr{C}}[i], M_{\mathrm{L}}[i]$ and $M_\gamma[i]$ the first, the second and the third component of the ith row of M, respectively.

Definition 15. Let (u, v) be a pair of primitive words and let γ be its coloring. The γ-*Colored Extended Burrows–Wheeler Transform* of S, denoted by $\mathtt{ebwt}(u, v)$, is the pair $(M_{\mathrm{L}}, M_\gamma)$.

Example 16. Consider the words $u = abaababb$ and $v = baabbaba$. In Fig. 1 we show the table M, containing, in the first column, the $\preceq_\omega$ sorted list of the conjugates of u and v and, in the second and third columns, the output of the `ebwt`. Then $\mathtt{ebwt}(u, v) = (bbbbbabaaababaaa, RBRBBRRBRBRBBRRB)$.

i	$M_{\mathcal{C}}$	M_L	M_γ
1	$aababbab$	b	R
2	$aabbabab$	b	B
3	$abaababb$	b	R
4	$abaabbab$	b	B
5	$ababaabb$	b	B
6	$ababbaba$	a	R
7	$abbabaab$	b	R
8	$abbababa$	a	B
9	$baababba$	a	R
10	$baabbaba$	a	B
11	$babaabab$	b	R
12	$babaabba$	a	B
13	$bababaab$	b	B
14	$babbabaa$	a	R
15	$bbabaaba$	a	R
16	$bbababaa$	a	B

Fig. 1. The table M corresponding to the transformation of the pair of words $u = abaababb$ and $v = baabbaba$ and with coloring $\gamma(u) = R$, $\gamma(v) = B$.

The colored ebwt transformation can be used in order to define a class of distance measures between words. Such measures are based on the remark that, when ebwt is applied to a pair of words (u, v), if the same factor s appears both in u and in v, then the conjugates of u and v starting by s are likely to be close in $M_{\mathscr{C}}(u, v)$. This implies that the greater the number of factors shared by u and v is, the greater is the mixing of conjugates of u and v in $M_{\mathscr{C}}(u, v)$. A distance measure based on the ebwt will take into account such a mixing. This intuition has different possible formalizations. One of these has been introduced by the authors in [27] (see also [29]) that considers the number of alternations of R and B in M_γ. Moreover, as shown in [26], the ebwt allows also to define a family of distance measures on words. All of the distance measures in this family formalize the intuitive idea that the greater is the number of factors shared by two sequences u and v, the smaller is the "distance" between u and v.

Let (u, v) be a pair of primitive words, let γ be its coloring and let $M_\gamma = M_\gamma(u, v)$. Let $x = M_\gamma[i]M_\gamma[i+1]\cdots M_\gamma[j]$ be a factor of M_γ. We define $n_x(u)$ ($n_x(v)$, respectively) as the number of characters colored by R (B, respectively) in the factor x of M_γ, that is

$$n_x(u) = \mathrm{card}\{l | i \leqslant l \leqslant j \quad \text{and} \quad M_\gamma[l] = R\},$$
$$n_x(v) = \mathrm{card}\{l | i \leqslant l \leqslant j \quad \text{and} \quad M_\gamma[l] = B\}.$$

We say that a sequence $\mathscr{P} = (x_1, x_2, \cdots, x_k)$ with $x_i \in A^*$ is a *parsing* of $w \in A^*$ if $w = x_1 x_2 \cdots x_k$. Every element x_j in the parsing is called a *block*.

Consider a pair of words (u, v) and the associated coloring function γ. Let $M_\gamma = M_\gamma(u, v)$ and let $\mathscr{P} = (x_1, x_2, \ldots, x_k)$ be a parsing of M_γ.

Definition 17. Let $u, v \in A^*$ be two primitive words and let $\mathscr{P}$ be a parsing of $M_\gamma(u, v)$. The *distance of u and v associated to the parsing* $\mathscr{P}$ is:

$$D_{\mathscr{P}}(u, v) = \sum_{x \in \mathscr{P}} |n_x(u) - n_x(v)|$$

In other words this definition means that for each parsing $\mathscr{P}$ of $M_\gamma(u, v)$, the distance $D_{\mathscr{P}}$ is defined as the sum of the differences of R's and B's into each block of the parse. Notice that this definition provides a family of distance measures, one for each parsing $\mathscr{P}$, as evidenced in the following example.

Example 18. Consider the transformation described by table M in Fig. 1. Then we can apply to M_γ the following parsing $\mathscr{P}$:

$$(RBRBBR)(RBRB)(RB)(BRRB)$$

In this case $D_{\mathscr{P}}(u, v) = 0$ since in each block there is an equal number of red and blue characters. If we consider a different parsing $\mathscr{P}'$:

$$(RBR)(BBR)(RB)(RB)(RB)(BRR)(B)$$

we get a different distance $D_{\mathscr{P}'}(u, v) = 4$.

Notice that all of the distance measures in this family have the *symmetric property*, that is for any pair of primitive words (u, v) and for any partition $\mathscr{P}$ of $M_\gamma(u, v)$, $D_{\mathscr{P}}(u, v) = D_{\mathscr{P}}(v, u)$. Instead it is not always true that if $u = v$, $D_{\mathscr{P}}(u, v) = 0$. In particular if $u = v$, then $D_{\mathscr{P}}(u, v) = 0$ if and only if all of the blocks in the parsing $\mathscr{P}$ have even length.

The converse is not always true. In fact we can find an even parsing $\mathscr{P}$ and a pair of different words u and v such that $D_{\mathscr{P}}(u, v) = 0$, as shows, for instance, the first parsing in Example 18.

An important property of our definition of distance, based on a parsing of the colored ebwt, is that it includes as particular case the k-tuple count Euclidean distance, reported in Section 2, which corresponds to a special choice of the parsing $\mathscr{P}$, depending on k. In fact in [26] it is proved that the number of occurrences of a given factor in u and v can be approximately detected by choosing a suitable factor in $M_\gamma(u, v)$. The theorem that we state in the sequel (see [26]) proves that we can find a suitable parsing $\mathscr{P}$ of $M_\gamma(u, v)$, depending on k, such that the corresponding measure $D_{\mathscr{P}}$ approximates the distance measure D_k. Recall that D_k is defined only for $k \leqslant \min\{|u|, |v|\}$. Remark that in practice the considered values of the resolution k are usually very

small (approximatively equal to 10) with respect to the lengths of u and v (several Kbytes). This ensures that the difference of such distances is very small, especially when one considers the normalized version of the distances, where the obtained values of $D_k(u,v)$ and $D_{\mathscr{P}(k)}(u,v)$ are divided by $|u|+|v|$.

Theorem 19. *Let the positive integer k be the resolution of the k-tuple distance D_k. Let u and v be two primitive words, let γ be the relative coloring and let $M_\gamma = M_\gamma(u,v)$. Then there exists a parsing $\mathscr{P}(k)$ of M_γ such that $|D_k(u,v) - D_{\mathscr{P}(k)}(u,v)| \leqslant 2(k-1)$.*

The underlying idea of the proof is the following. For a given k, the parsing $\mathscr{P}(k)$ of M_γ is obtained by grouping together in the sorted list $M_{\mathscr{C}}$ the elements having the same prefix of length k. So, for any length-k prefix x, one has a block $B(x)$ in the parsing $\mathscr{P}(k)$. Such a prefix corresponds to a length-k factor x occurring in u and v. Then we count the difference between the occurrences of x in u and v by looking at the difference between the occurrences of R and B in the corresponding block $B(x)$.

Notice that the approximation constant $2(k-1)$ corresponds to the fact that the prefixes of length k of the words in $M_{\mathscr{C}}$ include, besides all occurrences of the factors of length k of u and v, also the newly created factors because of conjugacy, i.e. the ones obtained as concatenation of a suffix of u (or v, respectively) with a prefix of u itself (v itself, respectively), that are not actually factors of the words u and v. This is illustrated in the following example.

Example 20. With reference to the table M of Fig. 1, the parsing $\mathscr{P}(3)$ of M_γ associated to the length $+3$ factors, is performed by considering the blocks corresponding to the elements in $M_{\mathscr{C}}$ having equal length 3 prefix

$$(RB)(RBBR)(RB)(RB)(RBBR)(RB).$$

Notice that each element in a given block corresponds to an occurrence of a given length-3 factor of u and v (cf. Fig. 1). For instance the first block (RB) correspond to the factor aab that has one occurrence in $u(R)$ and one occurrence in $v(B)$. The second block ($RBBR$) corresponds to the factor aba. However, in such a case, the length-3 prefix of the third element of $M_{\mathscr{C}}$ in this block does not correspond to an occurrence of the factor aba in u or v. It actually corresponds to the "new" factor created by concatenating the last a of v with the prefix ba of v itself. Then this new factor create a difference of 1 between $D_{\mathscr{P}(3)}(u,v)$ and $D_3(u,v)$. The overall distance of u and v associated to such a parsing is $D_{\mathscr{P}(3)}(u,v) = 0$ whereas $D_3(u,v) = 2$, as shown in Example 1. Instead, the parsing corresponding to the factors of length $+4$ is

$$(R)(B)(RB)(BR)(RB)(RB)(RBB)(R)(RB)$$

that is a refinement of the previous one. We have that $D_{\mathscr{P}(4)}(u,v) = 4$, whereas $D_4(u,v) = 8$.

The last example shows that, for any positive integer k, there exists a parsing $\mathscr{P}(k)$ such that $D_{\mathscr{P}(k)}$ approximates D_k. Such a $\mathscr{P}(k)$ can be considered a "good" parsing. However two problems arise: (1) Given M_γ, we do not have a criterion to determine "a priori" such a parsing $\mathscr{P}(k)$; (2) such a parsing $\mathscr{P}(k)$ is "good" relatively to k. But what is a "good" k?

The problem of choosing a "good" k is closely related to the identification of the optimal resolution mentioned at the end of Section 2. In the following we determine a very easily computable parsing of M_γ that captures several features of the parsing $\mathscr{P}(k)$ and, at the same time, is resolution-free, i.e. it is independent from the resolution k. Actually such an approach is able to use the optimal resolution even if such a value is not explicitly computed. Theoretical properties and experimental results (cf. [26]) show that actually such a parsing leads to a "good" distance measure. The choice of this parsing allows to overcome the limits of the distance D_k. The parsing is based on the following idea. As for the `bwt`, an important feature of `ebwt` is that it leads to group characters together so that the probability of finding a character close to another instance of the same character is substantially increased. In particular, given two words u and v having some structural similarity, if we consider a sufficiently long factor w of u and v, with a very high probability its occurrences in u and v are preceded by the same character a. This means that in $M_\gamma(u,v)$ the block of the letters preceding equal factors of u and v (that are close in the sorted list $M_{\mathscr{C}}$) is a monotonic block, i.e. a block composed only by the character a. Since our aim is to measure the differences of the frequencies of the factor w in u and v, respectively, we can approximately detect such a measure by looking at the different colors in the monotonic blocks in $M_\gamma(u,v)$.

More formally, if s is a word over an alphabet A, we say that a factor x of s is a *monotonic block* if x is made by equal characters (that is $x = a^k$ for some k) and it cannot be extended on the left or on the right in u with the same character. Let u and v be two primitive words and let $\mathtt{ebwt}(u,v) = (M_L, M_\gamma)$. Notice that every word can be decomposed in a unique way as concatenation of monotonic blocks. Consider the parsing of M_L into monotonic blocks. This parsing on M_L induces a corresponding parsing on M_γ. By extension, we define *monotonic blocks parsing* the parsing over M_γ above defined. The distance we are going to define considers the monotonic block parsing:

Definition 21. Let (u,v) be a pair of words, let $\mathtt{ebwt}(u,v) = (M_L, M_\gamma)$, and let $\mathscr{M}$ the monotonic block parsing of M_γ. The *monotonic block distance* is:

$$D_{BW}(u,v) = D_{\mathscr{M}}(u,v)$$

Example 22. Let $u = aaabbbb$ and $v = abaabbb$. It is easy to verify that $\mathtt{ebwt}(u,v) = (bbabaababbbbaa, RBRBBRRBBRBRBR)$. The monotonic block parsing of M_L is $(bb)(a)(b)(aa)(b)(a)(bbbb)(aa)$ that induces on M_γ the parsing $(RB)(R)(B)(BR)(R)(B)(BRBR)(BR)$ that leads to the distance $D_{BW}(u,v) = 4$.

The following proposition proves that our distance is a semi-metric, i.e. it is a positive measure that satisfies the symmetric property and the identity of indiscernibles.

Proposition 23. *Let u and v be two primitive words. Then $D_{BW}(u,v) = 0$ if and only if $u = v$.*

Nevertheless, we cannot state that D_{BW} is metric. In fact D_{BW} does not satisfy the triangle inequality, as shown in the following example:

Example 24. Consider the words $u = aaabbbb$, $v = abaabbb$ and $z = abababb$. It can be verified that $D_{BW}(u,z) = 4$, $D_{BW}(u,v) = 12$ and $D_{BW}(v,z) = 6$.

As remarked before, both of the distance measures D_{BW} and D_k provide values that are related to the number of factors shared by the considered strings. The formal connection between the ebwt-based distances with the distances based on the statistics of length k factors is given by Theorem 19. However the distance measure D_{BW} is resolution-free, so it overcomes the limit of the distance measure D_k that consists in the fact that the resolution k must be fixed "a priori". Actually, the distance D_{BW} captures the features of the distance D_k defined for the optimal resolution k.

The validity of the method is also confirmed by experimental results. The D_{BW} distance, in fact, experimentally shows to be a very good measure for mitochondrial genome phylogeny. The results of these experiments can be found in [26].

7. Conclusions

In this paper we have described different approaches for alignment-free sequences comparison, highlighting some relations between such different approaches. For instance Theorem 19 shows that, given a resolution k, we can find a suitable parsing $\mathscr{P}$ of the colored ebwt, depending on k, such that the corresponding measure $D_{\mathscr{P}}$ approximates the distance D_k. Moreover, experimental results (cf. [33,26]) show that there is a close relation between comparing sequences by D_{OS}, D_{BW}, and by some compression-based measures (cf. [10]). This seems to highlight that the needed information for comparison is hidden in what we get after the preprocessing phase of compression algorithms. As remarked in Section 5, in [15] it is proved that the distances D_{ZM} and D_{OS} are approximations of the Block Edit Distance D_B. It remains an open problem to prove whether D_{BW} is also an approximation of D_B. More generally, one can wonder whether there exists a suitable partition of the colored ebwt such that the corresponding measure is an approximation of D_B.

Moreover we remark that experimental results on biological sequences by using the D_{BW} distance measure, produce a clustering (cf. [26]) that agrees with the classification obtained with other methods. Hence we can guess that D_{BW} satisfies some weaker form of the triangle inequality. An interesting problem is to find a mathematical formalization of such a relation.

Acknowledgements

Partially supported by the Italian MIUR PRIN Project "Automi e Linguaggi Formali: aspetti matematici ed applicativi" and by MIUR FIRB Italy–Israel Project "Pattern Matching and Discovery in Discrete Structures, with applications to Bioinformatics".

References

[1] S.F. Altschul, W. Gish, W. Miller, E.W. Myers, D.J. Lipman, Basic local alignment search tool, J. Mol. Biol. 215 (1990) 403–410.
[2] D. Benedetto, E. Caglioti, V. Loreto, Language trees and zipping, Phys. Rev. Lett. 88 (4) (2002).
[3] C.H. Bennett, P. Gács, M. Li, P. Vitányi, W. Zurek, Information distance, IEEE Trans. Inform. Theor. 44 (4) (1998) 1407–1423.
[4] C.H. Bennett, M. Li, B. Ma, Chain letters and evolutionary histories, Sci. Am. (June) (2003) 76–81.
[5] B.E. Blaisdell, A measure of similarity of sets of sequences not requiring sequence alignment, Proc. Natl. Acad. Sci. USA 83 (1986) 5155–5159.
[6] B.E. Blaisdell, Effectiveness of measures requiring and not requiring prior sequence alignment for estimating the dissimilarities of natural sequences, J. Mol. Evol. (29) (1989) 526–537.
[7] M. Burrows, D.J. Wheeler, A block sorting data compression algorithm, Technical report, DIGITAL System Research Center, 1994.
[8] X. Chen, B. Francia, M. Li, Shared information and program plagiarism detection, IEEE Trans. Inform. Theor. 50 (7) (2004) 1545–1551.
[9] R. Cilibrasi, The complearn toolkit, 2003. http://complearn.sourceforge.net.
[10] R. Cilibrasi, P. Vitányi, Clustering by compression, IEEE Trans. Inform. Theor. 51 (4) (2005) 1523–1545.
[11] R. Cilibrasi, P. Vitányi, R. de Wolf, Algorithmic clustering of music based on string compression, Comput. Music J. 28 (4) (2004) 49–67.
[12] G. Cormode, S. Muthukrishnan, The string edit distance matching problem with moves, in: Proceedings of ACM-SIAM Symposium on Discrete Algorithms, 2001.
[13] G. Cormode, M. Patterson, S.C. Sahinalp, U. Vishkin, Communication complexity of document exchange, in: Proceedings of ACM-SIAM Symposium on Discrete Algorithms, 2000.
[14] M. Crochemore, W. Rytter, Jewels of Stringology, World Scientific, 2002.
[15] F. Ergun, S. Muthukrishnan, C. Sahinalp, Comparing sequences with segment rearrangements, in: Proceedings of the FSTTCS'03, Bombay, India, Lecture Notes in Computer Science, 2003, pp. 183–194.
[16] N.J. Fine, H.S. Wilf, Uniqueness theorem for periodic functions, Proc. Am. Math. Soc. (16) (1965) 109–114.
[17] D. Gusfield, Algorithms on Strings, Trees, and Sequences, Cambridge University Press, 1997.
[18] A. Lempel, J. Ziv, On the complexity of finite sequences, IEEE Trans. Inform. Theor. IT-22 (1) (1976) 75–81.
[19] A. Lempel, J. Ziv, A universal algorithm for sequential data compression, IEEE Trans. Inform. Theor. 23 (3) (1977) 337–343.
[20] A. Lempel, J. Ziv, Compression of individual sequences via variable-rate coding, IEEE Trans. Inform. Theor. 24 (5) (1978) 530–536.
[21] V.I. Levenshtein, Binary codes capable of correcting deletions, insertions, and reversals, Cybern. Control Theory 10 (8) (1966) 707–710, Original in Doklady Akademii Nauk SSSR 163(4) (1965) 845–848.
[22] M. Li, J.H. Badger, X. Chen, S. Kwong, P. Kearney, H. Zhang, An information based sequence distance and its application to whole mitochondrial genome phylogeny, Bioinformatics 17 (2001) 149–154.
[23] M. Li, X. Chen, X. Li, B. Ma, P. Vitányi, The similarity metric, IEEE Trans. Inform. Theor. 12 (5) (2004) 3250–3264.
[24] M. Li, P. Vitányi, An Introduction to Kolmogorov Complexity and its Applications, second ed., Springer, New York, 1997.
[25] D. Lopresti, A. Tomkins, Block edit models for approximate string matching, Theor. Comput. Sci. 181 (1) (1997) 159–179.
[26] S. Mantaci, A. Restivo, G. Rosone, M. Sciortino, A new combinatorial approach to sequence comparison (journal version), Theory of Computing Systems (TOCS), in press.
[27] S. Mantaci, A. Restivo, G. Rosone, M. Sciortino, An extension of the Burrows Wheeler Transform and applications to sequence comparison and data compression, in: Alberto Apostolico, Maxime Crochemore, Kunsoo Park (Eds.), Combinatorial Pattern Matching, 16th Annual Symposium, CPM 2005, Jeju Island, Korea, 19–22 June 2005, Proceedings, Lecture Notes in Computer Science, vol. 3537, Springer, 2005, pp. 178–189.
[28] S. Mantaci, A. Restivo, G. Rosone, M. Sciortino, A new combinatorial approach to sequence comparison, in: Mario Coppo, Elena Lodi, G. Michele Pinna (Eds.), Proceedings of ICTCS 2005, Theoretical Computer Science, 9th Italian Conference, ICTCS 2005, Siena, Italy, 12–14 October 2005, Lecture Notes in Computer Science, vol. 3701, Springer, 2005, pp. 348–359.
[29] S. Mantaci, A. Restivo, G. Rosone, M. Sciortino, An extension of the Burrows Wheeler Transform, Theor. Comput. Sci., in press.
[30] S. Muthukrishnan, S.C. Sahinalp, Approximate nearest neighbors and sequence comparison with block operations, in: Proceedings of ACM-STOC, 2000.
[31] S. Muthukrishnan, S.C. Sahinalp, Improved algorithm for sequence comparison with block reversal, in: Proceedings of LATIN, 2002.
[32] S.B. Needleman, C.D. Wunsch, A general method applicable to the search for similarities in the amino-acid sequences of two proteins, J. Mol. Biol. 148 (1970) 443–453.
[33] H.H. Otu, K. Sayood, A new sequence distance measure for phylogenetic tree construction, Bioinformatics 19 (16) (2003) 2122–2130.
[34] W.R. Pearson, D.J. Lipman, Improved tools for biological sequence comparison, Proc. Natl. Acad. Sci. USA 85 (8) (1988) 2444–2488.

[35] P.A. Pevzner, Statistical distance between texts and filtration methods in sequence comparison, Comput. Appl. Biosci. (8) (1992) 121–127.
[36] M. Rodeh, V.R. Pratt, Shimon Even, Linear algorithm for data compression via string matching, JACM 28 (1) (1981) 16–24.
[37] D. Sankoff, J.B. Kruskal, Time Warps, String Edits, and Macromolecules: The Theory and Practice of Sequence Comparison, Addison-Wesley Publishing Company, Reading, MA, 1983.
[38] J.S. Varre, J.P. Delahaye, E. Rivals, The tranformation distance: a dissimilarity measure based on movements of segments, Bioinformatics 15 (3) (1999) 194–202.
[39] S. Vinga, J. Almeida, Alignment-free sequence comparison – a review, Bioinformatics 19 (4) (2003) 513–523.
[40] R.A. Wagner, M.J. Fischer, The string-to-string correction problem, J. ACM 21 (1974) 168–173.
[41] J. Ziv, N. Merhav, A measure of relative entropy between individual sequences with application to universal classification, IEEE Trans. Inform. Theor. 39 (4) (1993) 1270–1279.